Technische Universität München
Institut für Informatik
Prof. Tobias Nipkow, Ph.D.
Brian Huffman, Peter Lammich

WS 2011/12
13. 01. 2012

Semantics of Programming Languages

Exercise Sheet 11

Exercise 11.1 Using the VCG, Total correctness

For each of the three programs given here, you must prove partial correctness and total correctness. For the partial correctness proofs, you should first write an annotated program, and then use the verification condition generator from *VC.thy*. For the total correctness proofs, use the Hoare rules from *HoareT.thy*.

A convenient loop construct:

```
abbreviation For :: "vname ⇒ aexp ⇒ aexp ⇒ com ⇒ com"
  ("(FOR _/ FROM _/ TO _/ DO _)" [0, 0, 0, 61] 61) where
  "FOR v FROM a1 TO a2 DO c ≡
     v ::= a1 ; WHILE (Less (V v) a2) DO (c ; v ::= Plus (V v) (N 1))"

abbreviation Afor :: "assn ⇒ vname ⇒ aexp ⇒ aexp ⇒ acom ⇒ acom"
  ("({_}/ FOR _/ FROM _/ TO _/ DO _)" [0, 0, 0, 0, 61] 61) where
  "{b} FOR v FROM a1 TO a2 DO c ≡
     v ::= a1 ; {b} WHILE (Less (V v) a2) DO (c ; v ::= Plus (V v) (N 1))"
```

Multiplication. Define an annotated program *MULTIPLY x y*, so that when the annotations are stripped away, it yields the program below. (The parameters x and y will appear only in the loop annotations.)

```
definition MULTIPLY :: "int ⇒ int ⇒ acom" where
lemma "strip (MULTIPLY x y) =
  (''c'' ::= N 0 ; FOR ''d'' FROM (N 0) TO (V ''a'') DO ''c'' ::= Plus (V ''c'') (V ''b''))"
```

Once you have the correct loop annotations, then the partial correctness proof can be done in two steps, with the help of lemma *vc_sound'*.

```
lemma MULTIPLY_correct:
  "⊢ {λs. s ''a'' = x ∧ s ''b'' = y ∧ 0 ≤ x} strip (MULTIPLY x y)
    {λs. s ''c'' = x*y ∧ s ''a'' = x ∧ s ''b'' = y}"
  by (rule vc_sound', auto simp add: MULTIPLY_def algebra_simps)
```

The total correctness proof will look much like the Hoare logic proofs from Exercise Sheet 9, but you must use the rules from *HoareT.thy* instead. Also note that when using rule *HoareT.While'*, you must instantiate both the predicate P :: *state* ⇒ *bool*

and the measure $f :: state \Rightarrow nat$. The measure must decrease every time the body of the loop is executed.

lemma *MULTIPLY_totally_correct*:
"$\vdash_t \{\lambda s.\ s\ ''a'' = x \wedge s\ ''b'' = y \wedge 0 \leq x\}$ *strip* (*MULTIPLY x y*)
$\{\lambda s.\ s\ ''c'' = x*y \wedge s\ ''a'' = x \wedge s\ ''b'' = y\}$"

Division. Define an annotated version of this division program, which yields the quotient and remainder of $''a''/''b''$ in variables $''q''$ and $''r''$, respectively.

definition *DIVIDE* :: "*int* ⇒ *int* ⇒ *acom*" **where**
lemma "*strip* (*DIVIDE x y*) = (
$''q'' ::= N\ 0$;
$''r'' ::= N\ 0$;
FOR $''c''$ *FROM* (*N 0*) *TO* (*V* $''a''$) *DO* (
$''r'' ::= Plus\ (V\ ''r'')\ (N\ 1)$;
IF Less (*V* $''r''$) (*V* $''b''$) *THEN SKIP*
ELSE ($''r'' ::= N\ 0$; $''q'' ::= Plus\ (V\ ''q'')\ (N\ 1)$))))"

Again, with the right annotations the partial correctness proof should be automatic.

lemma *DIVIDE_correct*:
"$\vdash \{\lambda s.\ s\ ''a'' = x \wedge s\ ''b'' = y \wedge 0 \leq x \wedge 0 < y\}$ *strip* (*DIVIDE x y*)
$\{\lambda s.\ x = s\ ''q'' * y + s\ ''r'' \wedge 0 \leq s\ ''r'' \wedge s\ ''r'' < y \wedge s\ ''a'' = x \wedge s\ ''b'' = y\}$"
by (*rule vc_sound'*, *auto simp add*: *DIVIDE_def algebra_simps*)

Also prove total correctness (replace $\vdash$ with $\vdash_t$).

Square roots. Define an annotated version of this square root program, which yields the square root of input $''a''$ (rounded down to the next integer) in output $''b''$.

definition *SQUAREROOT* :: "*int* ⇒ *acom*" **where**
lemma "*strip* (*SQUAREROOT x*) = (
$''b'' ::= N\ 0$;
$''c'' ::= N\ 1$;
WHILE (*Not* (*Less* (*V* $''a''$) (*V* $''c''$))) *DO* (
$''b'' ::= Plus\ (V\ ''b'')\ (N\ 1)$;
$''c'' ::= Plus\ (V\ ''c'')\ (V\ ''b'')$;
$''c'' ::= Plus\ (V\ ''c'')\ (V\ ''b'')$;
$''c'' ::= Plus\ (V\ ''c'')\ (N\ 1)$))"

Prove partial correctness using the VC generator, as shown.

lemma *SQUAREROOT_correct*:
"$\vdash \{\lambda s.\ s\ ''a'' = x \wedge 0 \leq x\}$ *strip* (*SQUAREROOT x*)
$\{\lambda s.\ s\ ''a'' = x \wedge (s\ ''b'')^2 \leq x \wedge x < (s\ ''b'' + 1)^2\}$"
by (*rule vc_sound'*, *auto simp add*: *SQUAREROOT_def power2_eq_square algebra_simps*)

Finally, prove total correctness of the square root algorithm.

Homework 11 Forward VCG

Submission until Wednesday, 25 January 2012, 12:00 (noon).

In the last tutorial, we have shown a forward assignment rule:

lemma *fwd_Assign*: "$\vdash \{P\}\ x::=a\ \{\ \lambda s'.\ \exists s.\ P\ s \wedge s'=s[a/x]\}$"
  apply (*rule hoare_relative_complete*)
  unfolding *hoare_valid_def*
  by *auto*

In this homework, your task is to implement a verification condition generator that uses forward reasoning.

fun *post* :: "*acom* ⇒ *assn* ⇒ *assn*"
fun *vc* :: "*acom* ⇒ *assn* ⇒ *bool*"
lemma *vc_sound*: "*vc c P* ⟹ ⊢ {*P*} *strip c* {*post c P*}"

Hint: Adapting the proof for the backward vcg from the lecture does not work well. You may have to show some cases manually, that are handled by *auto intro*: *conseq* in the proof from the lecture.

Note: You are only required to prove the soundness lemma. However, you should try to define your functions such that completeness also holds, i.e., we won't accept definitions like *vc* _ _ ≡ *False* or *post c P s* ≡ *True*.

lemma *vc_complete*:
  "$\vdash \{P\}c\{Q\} \Longrightarrow \exists c'.\ strip\ c' = c \wedge (vc\ c'\ P) \wedge (\forall s.\ post\ c'\ P\ s \longrightarrow Q\ s)$"

In order to test your vcg on programs, you may want to use the following corollary:

corollary *vc_sound'*:
  "$[\![(vc\ c\ P);\ (\bigwedge s.\ post\ c\ P\ s \Longrightarrow Q\ s)]\!] \Longrightarrow \vdash \{P\}\ strip\ c\ \{Q\}$"
by (*metis Hoare.weaken_post vc_sound*)